SIRAJ: A Unified Framework for Aggregation of Malicious Entity Detectors

Saravanan Thirumuruganathan, Mohamed Nabeel, Euijin Choo, Issa Khalil, Ting Yu
Qatar Computing Research Institute
{sthirumuruganathan,mnabeel,echoo,ikhalil,tyu}@hbku.edu.qa

***Abstract*—High-quality intelligence of Internet threat (e.g., malware files, malicious domains, phishing URLs and malicious IPs) are important for both security practitioners and the research community. Given the agility of attackers, the scale of the Internet, and the fast-evolving landscape of threats, one could not rely solely on a single source (such as an anti-malware engine or an IP blacklist) for obtaining accurate, up-to-date, and comprehensive threat analysis. Instead, we need to *aggregate* the analysis from multiple sources. However, it is non-trivial to do such aggregation effectively. A common practice is to label an indicator (malware, domains, URLs, etc.) as malicious if it is marked by a number of sources above an ad-hoc certain threshold. Often, this results in sub-optimal performance as it assumes that all sources are of similar quality/expertise, independent, and temporally stable, which unfortunately are often not true in practice. A natural alternative is to train a supervised machine learning model. However, this approach needs a sufficiently large amount of manually labeled ground truth, which is time-consuming to collect and has to be updated frequently, resulting in substantial recurring costs.**

In this paper, we propose SIRAJ, a novel framework for aggregating the detection output of various intelligence sources such as anti-malware engines. SIRAJ is based on the pre-train and fine-tune paradigm. Specifically, we use self-supervised learning-based approaches to learn a pre-trained embedding model that converts multi-source inputs into a high-dimensional embedding. The embeddings are learned through three carefully designed pretext tasks that imbue them with knowledge about dependencies between scanners and their temporal dynamics. The learned embeddings could be used for diverse downstream machine learning tasks. SIRAJ is designed to be general and can be used for diverse domains such as URLs, malware, and IPs. Further, SIRAJ works well even when there is limited to no labeled data available. Through extensive experiments, we show that our learned representations can produce results comparable to supervised methods while only requiring as little as 100 labeled samples. Importantly, the results show that SIRAJ accurately detects threat indicators much earlier than the baseline algorithms, a feat that is critical against short-lived indicators like Phishing URLs.

I. Introduction

High-quality intelligence of Internet threat (aka, threat vectors) such as malware files, malware URLs (that are used for malware distribution), and malicious IPs, is vital to both security practitioners and researchers. For practitioners, such intelligence will help detect and prevent compromises, and effectively protect users and critical IT infrastructure. For researchers, high-quality intelligence is paramount for reliable and comprehensive experimental evaluation of novel machine learning-based cyber security solutions. Considering the ever-evolving landscape of Internet security, one cannot rely on a single source for detecting threats. An alternative is to *aggregate* the output from multiple sources and then integrate them to derive a more reliable score. VirusTotal [1], for example, is a popular website that provides scanning services for IPs, files, and URLs. For concreteness, let us consider the VirusTotal URL scan service for detecting phishing and malware-hosting URLs. It aggregates results from over 70 third-party scanners ranging from popular ones such as Google Safe Browsing (GSB), Sophos, and Fortinet to less known ones such as Lumu, VX Vault, and GreenSnow. These scanners exhibit different types of expertise – for example, OpenPhish specializes in detecting phishing URLs, while URLhaus specializes in detecting malware URLs, and GSB specializes in both. This divergence makes the problem of combining multiple-source intelligence (such as VirusTotal) into an integrated high-quality one quite challenging.

Prior Approaches and Their Limitations. Given the importance of this problem, there has been extensive work on developing various heuristics for aggregating the predictions of multiple detectors into a single detection. A simple and commonly used method is to use a *cutoff threshold* τ. If more than τ intelligence sources (e.g., VirusTotal scanners) identify an entity as malicious, it is treated as malicious. However, there is no consensus among the community on the appropriate threshold. For example, prior work using VirusTotal has chosen thresholds of 1 [2], [3], [4], 2 [5], [6], and 5 [7]. Setting a too-high threshold would miss a lot of malicious cases, while a too-low threshold would introduce many false positives. The threshold-based approach implicitly treats each intelligence source as having similar expertise and quality, contradicting several prior works [8], [9], [10], [11]. Furthermore, using a static threshold cannot handle dynamic scenarios where the quality of intelligence sources varies with time. This could result in fluctuations of the maliciousness of an entity which is undesirable [8], [9]. An alternate approach is to train a machine learning (ML) model that can take as input the predictions of multiple sources (such as VirusTotal URL scan reports or multiple IP blacklists) and output a single prediction about the maliciousness of the entity. The state-of-the-art approaches [12], [5], [13], [14], [15], [16], [17], [18], [19], [20] have achieved high accuracy. However, such approaches are not sustainable in practice as training ML models (especially deep learning based) require a large number of manually labeled training data. Collecting such data is time

consuming and requires constant updates to compensate for concept drift where the distribution of the test data increasingly differs from the training data.

A. Outline of Technical Results

We propose SIRAJ, a novel solution framework that can intelligently and accurately aggregate the predictions of various threat detectors. SIRAJ has the following properties that distinguish it from prior work. SIRAJ is a unified framework that can be applied for diverse domains in a transparent manner. In our experiment section, we evaluate our approach over four types of malicious entities – Phishing URLs, malware URLs, malware files and malicious IPs. SIRAJ achieves this by a careful design of domain-agnostic representations. In other words, instead of designing features for each domain (URLs vs. malware files vs. IPs), we propose a novel approach that represents prediction of multiple sources as a dense high-dimensional vector that encodes various information such as the dependencies between different sources and their temporal dynamics. Another practical concern is that the initial threat indicators are often not reliable, and it takes several hours or even days before they produce stabilized results for fresh inputs. However, phishing and malware attacks increasingly use disposable URLs used only for a few hours. To reduce the potential damage, it is paramount to detect malicious entities from early unreliable reports. SIRAJ's embeddings capture the temporal relationship among different intelligence sources, allowing it to detect malicious entities much earlier than either threshold-based or supervised ML-based approaches.

Overview of Our Approach. Our framework is based on the pre-train and fine-tune paradigm [21]. It is often much easier to collect a number of *unlabeled* reports from sources such as VirusTotal and aggregated IP blacklists. However, the cost of labeling the data is much higher. Hence, it is desirable to design a solution that requires limited to no labeled data. We design *three* novel pretext tasks for which labeled data could be obtained *automatically* without the need of any domain expert. These tasks are carefully designed to learn relevant features such as the dependencies between various sources and their temporal dynamics. The output of these tasks is an encoder that can convert multiple intelligence reports into an embedding that is domain and time-invariant. We use multi-task learning [22] so that we learn a single encoder for all these three tasks. Once the encoder is pre-trained using unlabeled data, it could then be used for diverse downstream tasks (e.g., classification, clustering or prediction). To the best of our knowledge, we are the first to propose such a generic framework based on self supervised learning that could be reused for multiple types of malicious entities. We focus on two common settings – one with a limited amount of labeled data and one with no labeled data at all. Our learned representations can outperform supervised methods while only requiring as little as 100 labeled samples.

Summary of Contributions.

- We propose SIRAJ, a unified framework for aggregating multi-source intelligence for diverse entity types, including URLs, malware files, and IPs.
- We design three novel pretext tasks for self-supervised learning. The goal of these tasks is to learn the dependencies between intelligence sources and temporal dynamics across time. We learn a single embedding that is consistent with all these three tasks.
- SIRAJ outperforms threshold-based, supervised ML-based, and generative model-based approaches, in terms of not only accuracy but also timely intelligence. Our embeddings are robust to random and adversarial corruptions and gracefully handle concept drift.

Paper Outline. We introduce the relevant preliminaries in Section II. We provide an overview of the components of SIRAJ in Section III. The generative model is introduced in Section IV while the pretext tasks for self-supervised learning are introduced in Section V. We describe how to use SIRAJ to develop supervised, semi-supervised and unsupervised classifiers for diverse downstream tasks in Section VI. We describe experimental results in Section VII, and conclude with some parting thoughts in Section IX.

II. Preliminaries

SIRAJ is designed to work with the intelligence of diverse Internet threats. In this paper, we use the generic term *entity* to refer to objects that could be either benign or malicious and be used in attacks, e.g. binaries, domains, URLs and IPs. We use the term *scanner* to represent a source of threat intelligence of a certain type of entity, e.g. an antivirus engine, an IP blacklist or a domain reputation system. When queried about an entity e, a scanner would report its own assessment of e, which, without loss of generality, could be malicious, benign or no information. The assessment from all scanners together form the *scan report* of the entity e. We use the term *label* to denote the maliciousness or benignness of an entity. Formally, let $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ be the set of m entities. Suppose there are n scanners $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. We represent the input dataset for intelligence aggregation by a matrix $X = (X_{ij})$ $1 \leq j \leq m$ and $1 \leq j \leq n$ with

$$X_{ij} = \begin{cases} 1, & \text{if } s_j \text{ detected } e_i \text{ as benign} \\ -1, & \text{if } s_j \text{ detected } e_i \text{ as malicious} \\ 0, & \text{if } s_j \text{ did not scan } e_i \end{cases}$$

The scan report R_e for an entity e is a vector of dimension n containing the assessment from each scanner $s \in \mathcal{S}$ for e. For a collection of entities $e \in \mathcal{E}'$ where $\mathcal{E}' \subseteq \mathcal{E}$ we also collect a time series of scan reports over a period of time $\mathcal{T} = \{T, T + \delta, T + 2\delta, \ldots\}$ where T represents the starting time and δ represents the periodicity of data collection such as 6 hours, 1 day, etc. R_e^t is the scan report for entity e at time t. We omit t when the context is clear. We use the time series to understand the temporal dynamics of scan reports.

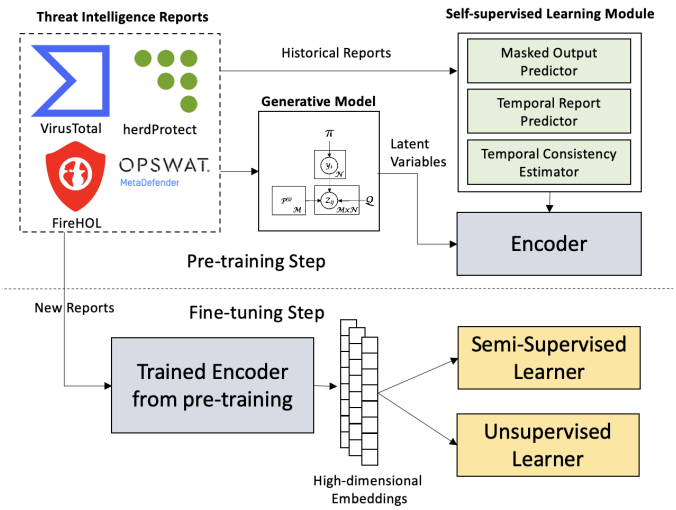


Fig. 1: Overview of Our Approach

III. Overview of Our Approach

SIRAJ synthesizes diverse ideas from machine learning and we provide a high level overview of its key components in this section. Figure 1 illustrates the general overview of our approach. Our solution can be decomposed into three key steps: (1) Generative models, (2) Pre-training, and (3) Fine-Tuning. We provide a rationale for each of the components and how all these components fit together in the following. Then, we provide additional details in Section IV, Section V, and Section VI.

Solution Framework. Our goal is to learn an appropriate intermediate embedding representation for scan reports, which could be used for downstream tasks such as detecting malicious entities without any labeled data or with minimal labeling. To the best of our knowledge, we are not aware of any prior work that seeks to learn such an embedding. We emphasize that our approach is generic and customizable. The same approach could be used for diverse malicious entities such as malware, URLs, and IPs through VirusTotal scan reports or IP blacklists.

1. Generative Models. Prior work such as [10] has shown that understanding the dependencies between the scanners (using a generative model) is useful for aggregating the scanner outputs. However, directly using generative models often fails due to the mismatch between modeling assumptions and the real-world. Hence, instead of using the generative model for *prediction*, we use it to learn the *dependencies* and *dynamics* exhibited by the scanners and their labels. The learned model is used to inject domain knowledge into the self-supervised learning tasks.

2. Pre-Training using Pretext Tasks. While the generative model is a promising start, it is insufficient on its own for accurate aggregation of scan reports. The generative model is an unsupervised approach that can learn the dependencies at the *corpus* level. It is necessary to learn more granular details about the scanner dynamics for accurate aggregation. We achieve this by employing a self-supervised learning approach with careful design of three inter-related pretext tasks that imbue the embedding model with knowledge about dependencies between scanners and their temporal dynamics. The first task seeks to learn the dependencies between the scanners when generating the scan report for an entity. While the first task superficially looks similar to the generative model, they both learn the scanner dependencies at different granularities. For example, GM might learn that scanners S_i and S_j have high correlation in their reports. On the other hand, task 1 model could learn that the output of S_j for a particular entity could be predicted from the output of S_i, S_k and S_l. Later, we conduct experiments to showcase the complementarity of the generative model and the first task. The second task tackles the disconnect between early and late scan reports of a new entity (e.g., phishing or malware URL) by modeling the temporal dynamics of scanner responses. The goal of the third task is to learn embeddings that are temporally consistent. SIRAJ uses a multi-task approach to learn embeddings that are simultaneously well suited for all pretext tasks.

3. Fine-Tuning. Once the encoder is trained from pretext tasks, it can take any scan report and generate an embedding that could be used for various downstream tasks such as detection and clustering. We consider two fine-tuning scenarios. First, when there is a limited number of labeled data available (as little as 100 scan reports), we use a semi-supervised approach to train an effective discriminative model that achieves similar performance as a supervised model trained over a much larger labeled corpus. Second, when there are no labeled data available, we leverage the properties of the embedding space to design an unsupervised classifier that outperforms a wide variety of other unsupervised approaches such as heuristic thresholds, generative models, and weak supervision.

IV. Generative Models for Aggregation

In this section, we describe our approach to learn the dependency structure of a generative model and use it to model the latent variables of the scanners.

Generative Modeling for Aggregation. A principled approach for aggregating unlabeled data is generative models. The key insight is to model the process by which the unlabeled data (i.e., scan reports) was generated. We treat the true label of a scan report (malicious or benign) as a latent variable that generates the observed and possibly noisy scan report. This contrasts with traditional approaches that try to model the true label given the noisy scan report. Once the parameters of an appropriate generative model have been fitted using the unlabeled data, we could use it to estimate the latent true label of a scan report. This approach is counter intuitive - yet yields accurate results if the generative model is appropriate.

Learned vs Specified Generative Models. Prior works that applied generative models for VirusTotal aggregation [10], [23] assume that generative models are pre-specified by the domain expert and only focus on estimating the latent variables. However, the structure of the generative model has

a significant impact on its accuracy. The biggest source of inaccuracy is ignoring the statistical dependencies between the scanners. For example, scanner s_1 and s_2 could have highly correlated scan reports for different entities as they use the same algorithm in the back end [8]. Not taking that into account could seriously affect the accuracy of aggregation. It is unreasonable to expect even a domain expert to specify the generative models with accurate dependencies.

In this work, we advocate for a novel two-step approach. First, we leverage recent innovations in weak supervision [24] to efficiently learn the dependencies. Second, we specify the generative model using the learned dependencies and use it to estimate the latent variables corresponding to scanner accuracies. Our approach differs from prior work in two aspects: (a) our generative model is constructed using a data-driven approach; (b) we do not directly use the estimated latent variables for aggregation – instead, we use them to infuse the pretext tasks with appropriate domain knowledge.

Learning Dependency Structures. Our generative model for learning the dependencies between scanners is based on a factor graph [25]. Such graphs consist of two types of nodes – *variable* nodes corresponding to scanners and their outputs and *factor* nodes that define the relationships between variable nodes. For example, the algorithm could designate two scanners s_i and s_j as related by creating a new factor node f_k connecting to both s_i and s_j. This simple graph structure is expressive enough to model arbitrary relationships, including high order dependencies. Similar to [26], we consider pairwise and conjunctive dependencies. The key challenge is that the number of dependencies explodes with the increasing number of scanners. For example, even if we limit ourselves to pairwise dependencies between any pair of scanners, there are more than 2500 such dependencies given VirusTotal's over 70 scanners. One needs a huge amount of data to identify which of these potential dependencies are irrelevant reliably. We rely on prior works for efficiently learning the generative model. First, we learn the dependency structure by leveraging the techniques from [26] that proposed an optimization formulation for estimating the log marginal pseudo-likelihood of the output of a single scanner conditioned on the outputs of all other scanners. Second, once the dependency structure is known, we use [27] that breaks the generative model into smaller sub-problems and learns the parameter values through closed-form solutions. This approach allows one to learn the parameters in time linear in the size of the data, which is appealing in a domain where the unlabeled data is abundant.

V. Self Supervised Learning based Approach

Self supervised learning (SSL) is a recently emerged technique that builds a supervised learning task (i.e., a pretext task) from the dataset itself, not from the manually annotated data [28]. SSL automatically generates pseudo labels based on the attributes in the data such that pseudo labels are correct for a specific task. Then model is trained in a supervised manner using the generated pseudo labels. Essentially, SSL works by designing *pretext tasks* to be solved during a pre-training step. The model trained to solve those pretext tasks learns representations of data that will be used for downstream tasks.

SSL has been extensively studied for computer vision and natural language processing. These domains are well suited for SSL due to the spatial and semantic structure present in image or language data. However, such properties do not exist in tabular domain data such as scan reports, making the problem of data augmentation and SSL challenging [28]. To the best of our knowledge, we are not aware of any work on SSL for aggregating multi-source intelligence such as VirusTotal. We introduce a novel SSL approach to learn domain and time invariant representations that are useful for diverse downstream tasks.

Desiderata for Pretext Task Design. A key challenge in SSL is the design of an appropriate pretext task such that (a) the knowledge/embeddings learned is relevant for the downstream task; and (b) the pseudo-labels for pretext tasks can be generated *automatically* and efficiently. We desire that the embeddings are generic enough to be used for training ML models for a broad spectrum of downstream tasks such as (early) detection of malicious entities and detection of attack types. This could only be achieved by learning some intrinsic properties of the scan reports.

Transferable Properties of Scan Reports. There has been extensive work on analyzing the properties of VirusTotal scan reports along different dimensions such as relative accuracy, stability, and convergence [29], [8], [9]. Our analysis of these works identified some generic properties across various domains and multi-source intelligence aggregators.

- Not all scanners are equal. They vary in their accuracy, expertise, and consistency of their responses.
- Scanners have sophisticated dependencies between them. Some scanners have a higher degree of overlap in their responses than others due to various reasons such as shared expertise. Some scanners could have specialized expertise and respond only to a specific class of entities such as Phishing URLs.
- Complex *temporal* dependencies exist between scanners. Even for a fixed entity e, individual scanners could have disparate behavior over a period of time. Some scanners could produce a detection result (malicious or benign) and stick to it. Alternatively, some scanners could *flip* their detection results intermittently. Scanners for malware often exhibit hazardous flips [8] where the label flips to a different value twice in a short period of time (such as malicious to benign and back to malicious). Similarly, some scanners are conservative and lag behind others, while other scanners could be bellwethers.

Our goal is to design pretext tasks that are cognizant of these behaviors. It is challenging to design a *single* pretext task that can address all the relevant desiderata. Instead, we propose *three* intuitive tasks that jointly learn the necessary dependencies using a multi-task learning framework. Figure 2 illustrates our approach where each row represents each task.

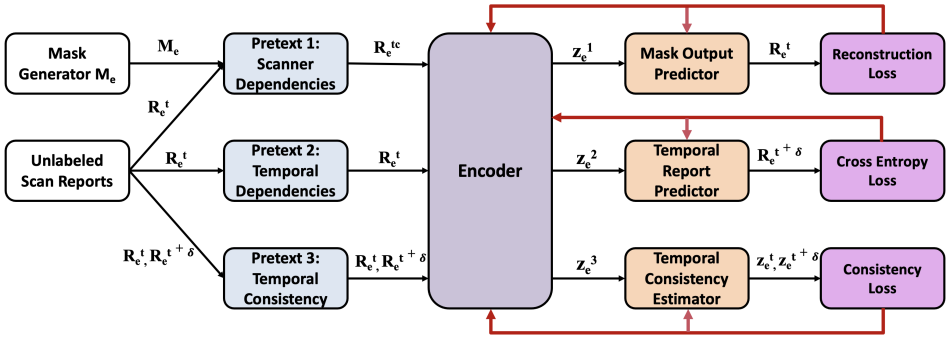


Fig. 2: Illustration of Self Supervised Approach.

Given a scan report R_e, we pass it to pretext generators for three pretext tasks, respectively. *Pretext Generator 1* corrupts the scan report based on the masks, while *Pretext Generator 2* and *Pretext Generator 3* function as identity functions by passing the input without modification. The corrupted and original scan reports are passed to an encoder to generate embeddings z_e^1, z_e^2, z_e^3 that are then fed to the task specific predictors. We then apply the pretext task-specific loss function on the predicted output and apply back propagation to improve upon both the encoders and the predictors. In the following, we describe each of three tasks in detail.

A. Task 1: Scanner Dependencies

Given an unlabeled dataset of scan reports for various entities, the goal of the first pretext task is to learn the high-level dependencies between scanners. A simple but intractable approach would be to collect some summary statistics such as correlations between scanners. For example, even if we limit ourselves to pairwise correlations, there are more than 2500 such values for the 70-odd scanners from VirusTotal's URL scanning service. Including higher-order correlations increases this number exponentially. Hence, an alternate and more tractable reformulation is needed.

Our key insight is to formulate this as a self-supervised learning problem. We are given a dataset of scan reports for m entities over n scanners. Let $p \in (0, 1)$ be a hyper-parameter. Given a scan report R_e for entity e of dimensionality n (one entry per scanner), *Mask Generator* generates a binary mask vector M_e of dimensionality n where each component takes the value of 1 with probability p. For example, if $p = 0.1$ and $n = 80$, then we would expect 10% of the mask vector M_e to be 1. Given M_e, *Pretext Generator1* transforms the scan report R_e such that whenever $M_e[i] = 1$, $R_e[i]$ is corrupted by 'swap noise' – i.e., we replace it with the corresponding entry from $R_{e'}[i]$ where $R_{e'}$ is a randomly chosen scan report from the unlabeled dataset. Let R_e^c be the corrupted version of R_e based on M_e. We design a pretext task that can output the original scan report R_e from the corrupted scan report R_e^c. Note that we could generate virtually unlimited labeled data by generating multiple corrupted versions for each scan report R_e. The learning is performed in two phases. First, we pass R_e^c to an encoder that outputs an embedding z_e^1. Next, we train a predictor (*Mask Output Predictor*) that takes z_e^1 as input and outputs R_e. We perform joint learning where the back-propagation improves both components. We can see that to do well on this pretext task, the SSL has to design an appropriate encoder and also learn the various dependencies between the scanners. For example, assume that scanners s_i, s_j, s_k exhibit high correlation between their responses and $R_e[i]$ is corrupted. Then the predictor could learn to fix it through the dependencies with $R_e[j]$ and $R_e[k]$.

B. Task 2: Temporal Scanner Dependencies

Pretext task 1 takes a single scan report R_e and learns the dependencies within them. The goal of pretext task 2 is to learn the dependencies across time. Suppose that R_e^t and $R_e^{t+\delta}$ are the scan reports for the same entity e for time t and $t + \delta$. We design an SSL task that takes R_e^t as input and produces $R_e^{t+\delta}$ as output. In other words, we seek to predict the 'future' scan report from the current report. This task is useful for two reasons. First, while it is often challenging to determine whether an entity e is malicious from an early scan report, it is usually much easier after a certain period has elapsed (dubbed the stabilization period) [8]. Second, this pretext task results in the predictor learning both the scanner and label dynamics. If a scanner often flip-flops or lags behind another, it will be learned by the predictor. Once again, we use a two-step process where we pass R_e^t to the encoder to get z_e^2 that is then passed to the predictor (*Temporal Report Predictor*) with an expected output of $R_e^{t+\delta}$.

δ is a hyper-parameter and depends on the domain at hand. The granularity of δ could be in hours for fast moving domains such as Phishing URLs while it could be in days for other domains. One may obtain the value of δ by analyzing a time series of scan reports and identifying the appropriate periodicity that captures major inflection points. For example, [8] finds that after a malware file is submitted to VirusTotal for four weeks, the scanners' labels become stable and do not exhibit any hazardous flips. Hence, the value of δ should be much lower than four weeks. Similarly, works such as [9] and our analysis for VirusTotal reports for URLs show a delay Δ in updating VirusTotal labels due to its non-proactive pull method. Hence, one should set $\delta \geq \Delta$.

C. Task 3: Temporal Prediction Consistency

Let $R_e^t, R_e^{t+\delta}, R_e^{t+2\delta}, \ldots$ be a time series of scan reports for the same entity e. For example, these could be the reports for the same entity taken every $\delta = 6$ hours. Recall that pretext task 1 learns the scanner dependencies, and pretext task 2 seeks to predict the future scan report from the current one. A useful additional constraint is to ensure that the embeddings for the same entity across consecutive timestamps must be sufficiently similar so that a predictor is likely to make the same prediction for each of the scan reports in the time series. This constraint is motivated by the fact that the benignness/maliciousness of an entity does not change as long as δ is relatively small. If the entity is malicious, we wish to learn embeddings such that the predictor can provide consistent predictions for that entity across the time series. Similarly, if the entity is benign, we would expect the predictor to give the same label consistently across the time series. By ensuring that the embeddings are similar across time, the predictor that we train will be able to provide accurate and early predictions. This is achieved through consistency loss that penalizes two temporally consecutive embeddings for $R_e^{t+\delta}, R_e^{t+2\delta}$ based on their Euclidean distance. Embeddings that are farther are penalized more than those that are closer together.

D. Embedding Harmonization via Multi-Task Learning

In a typical SSL setting, one designs a single pretext task and uses the learned encoder for computing the embeddings. However, the complexity of the cyber security domain in terms of scanner dependencies and temporal dynamics precludes this simple approach. Instead, as discussed before, we design three inter-related pretext tasks, each of which focuses on different yet relevant facets of the problem. Each of these tasks takes a scan report R_e as input and uses an encoder to output an embedding z_e that is used by the predictor of the individual pretext task. This creates a new challenge of harmonization where we strive to constrain the individual encoders to learn embeddings that are useful across the tasks.

This is achieved through the use of multi-task self-supervised learning. We seek to learn each of these three tasks simultaneously and force the individual encoders to share the knowledge across these tasks. Intuitively, we learn a shared representation between the tasks enabling it to generalize better by ignoring data-dependent noises. Even though the tasks are sufficiently different, they also share several commonalities that result in increased efficacy of learning shared representations and thereby achieve superior prediction accuracy than training task-specific models individually. We use the same model architecture for the encoder for all three tasks. We use *soft* parameter sharing where each of the encoders has its own internal parameters. However, we try to minimize the ℓ_2 distance between the parameters to encourage them to be similar. The parameters of the encoders are weakly tied based on regularization so that they do not stray widely. Overall, the self-supervised training is performed without any labeled data resulting in a generic encoder that takes a scan report and produces an embedding.

VI. Putting It All Together

In Section IV, we propose a generative model that learns the latent variables for each scanner. In Section V, we design self-supervised learning tasks to learn effective embeddings. In this section, we describe how to use these learned embeddings to tackle various downstream problems. For the sake of concreteness, we focus on the problem of malicious entity detection. Given a scan report R_e, our goal is to output whether the entity e is malicious or benign as early as possible.

We focus on two key scenarios – one where there is *limited* labeled data available, and another with *no* labeled data available. Overall, our solution approach is based on the pre-train and fine-tune paradigm. We use the generative model and the SSL tasks to learn a good encoder in the pretraining step. In the fine-tuning step, we leverage the encoder and its embeddings for early and accurate detection.

A. Scenario 1: Limited Labeled Data

As we shall show in the experiments, it is possible to reduce the required amount of labeled data using the proposed embeddings as they encode some domain knowledge. We consider a scenario with insufficient labeled data to train a fully supervised model either on the original scan reports or their embedding counterparts. This falls under the realm of semi-supervised learning, where the goal is to train a more generalizable classifier than one that is trained only on the labeled data. Our proposed approach is based on [28].

Let D_L be the set of labeled data. Let $R_e \in D_L$ be a scan report, and y_e be the corresponding label (such as whether e is malicious or benign). For each $R_e \in D_L$, we generate K masks $m_e^1, m_e^2, \ldots, m_e^K$ using the mask generator from pretext task 1 with $p = 0.05$ – in other words, 5% of the mask vectors will have a value of 1 and will cause corruption of R_e. We corrupt R_e using the masks to obtain $\mathcal{R}_e^c = \{R_e^{c1}, R_e^{c2}, \ldots, R_e^{cK}\}$. Our key insight is that the label for both R_e and each of the corruptions have to be the same. Hence, we pass the original and corrupted scan reports, R_e and $\mathcal{R}_e^c$, to the encoder to get the respective embeddings z_e and $\mathcal{Z}_e^c = \{z_e^{c1}, z_e^{c2}, \ldots, z_e^{cK}\}$. We train a predictor by passing z_e and $\mathcal{Z}_e^c$ with y_e as the expected output for each of them. Let the predictions be $\tilde{y}_e$ and $\tilde{\mathcal{Y}}_e^c = \{y_e^{c1}, y_e^{c2}, \ldots, y_e^{cK}\}$. We use cross entropy loss between the true label y_e and the predicted label $\tilde{y}_e$. We apply the consistency loss between $\tilde{y}_e$ and $\tilde{\mathcal{Y}}_e^c$ which penalizes when the predictions for the original scan report and its corruptions diverge. We then update the parameters of the predictor by back-propagating both the supervised and consistency loss. This process is repeated for each $R_e \in D_L$. In summary, our solution leverages both the learned embeddings and the first pretext task to learn a supervised classifier. We observe that our classifier with as little as 100 labeled data achieves the same performance as a fully supervised classifier requiring an order of magnitude more data.

B. Scenario 2: No Labeled Data

This is a challenging scenario where we have to perform the detection of malicious entities without any labeled data. We rely on the following insight – while it is hard to predict the maliciousness of an entity based on the early scan reports, it is often much easier to predict them based on the scan reports after the stabilization period [8]. In fact, one could use relatively simple predictors to achieve high accuracy for the late stage scan reports. While there have been a handful of unsupervised models for detecting malicious URLs [10], [23], they often do not achieve good results for early detection. We address this conundrum through the predictor for the second pretext task that can generate the future scan report.

Let R_e^t be the current scan report while $R_e^{t+\delta}$ be the output of the predictor of the second pretext task. Let z_e^t and $z_e^{t+\delta}$ be their corresponding embeddings. Let $\tilde{z}_e$ be the concatenated vector $[z_e^t, z_e^{t+\delta}]$. As a pre-processing step, we compute the concatenated embeddings $\tilde{z}_e$ for all entities. Let $\mathcal{Z}$ be the set of concatenated embeddings for all entities.

Training an unsupervised model only based on R_e^t or z_e^t would result in sub-optimal performance. Instead, we follow a three-step process for boosting knowledge transfer from pretext tasks inspired by [30]. First, we cluster all the vectors $\tilde{z}_e \in \mathcal{Z}$ into *two* clusters that intuitively correspond to malicious and benign, respectively. Let C_1 and C_2 be the cluster centroids. Second, for each $\tilde{z}_e$, we assign it to one of the clusters based on their proximity to C_1 and C_2. Finally, we train a new predictor that takes embedding as input and outputs the cluster assignment. Specifically, the model is trained over the set $\{(z_e^t, c_e)\}$ where z_e^t is the embedding of the current scan report R_e^t and c_e is the cluster assignment. In other words, we train a supervised model for predicting the cluster that an embedding must belong to. Our experimental results and a number of prior work including [30] show that this approach works well in the presence of an appropriately trained encoder which is the case in our setting. Once we obtain the two clusters, we could identify the malicious cluster in one of two ways. If one expects the number of benign entities to be larger than that of malicious entities, then the larger cluster would correspond to the benign entity cluster. Alternatively, we could sample a small number of entities (such as 1-5) from either of the clusters and verify them with some external repository (such as PhishTank or URLhaus) or obtain the appropriate label from the domain expert.

VII. Experiments

We conduct extensive experiments to showcase the generality and efficacy of SIRAJ. The code for SIRAJ can be found at https://github.com/qcri/SIRAJ.

A. Data Collection

In contrast to prior works that focus on a particular type of entities, the generality of our approach enables us to conduct experiments over four types of entities – phishing URLs, malware URLs, malware files, and blacklisted IPs. The statistics of the datasets can be found in Table I. Recall that an entity may have multiple scan reports corresponding to different scanning time points.

Malware Files. We use the data collected from [8] for our experiments. This data consists of two partitions. The 'main' dataset contains 14,423 files and their daily labels of 65 scanners from VirusTotal over a period of one year (unlabeled main). The authors submitted files that were 'fresh' and were submitted to VirusTotal for the first time. The 'auxiliary' dataset consists of 356 files collected that were manually verified by the authors. We use the unlabeled main dataset to train our generative and self-supervised models and use the auxiliary dataset for fine-tuning and evaluation.

Phishing and Malware URLs. We collect three different datasets for pre-training, fine-tuning and final evaluation. Our data collection process is inspired by the data collection procedure in [8] for malware.

Pre-Training. Malicious URLs are often short lived as it is economical to create new URLs [33]. Hence, it is important to track the change in scan reports much more frequently. Instead of using the 1-day granularity for malware collection, we collect hourly scan reports for URLs. Our institution has subscribed access to VirusTotal URL feed, which contains all the URLs submitted to VirusTotal every day along with the submission timestamps and the corresponding scan reports. This unlabeled data is used to train the generative and self-supervised models for both phishing and malware URLs.

Fine-Tuning. For phishing URLs, we use PhishTank (PT), a collaborative website for verifying Phishing URLs. Users submit suspected Phishing URLs to PT that are then verified and voted upon by other users. We implement a crawler to collect the newly submitted URLs to PT. We filter out invalid URLs (e.g., malformed URLs) and then immediately submit them to VirusTotal, and collect the scan reports. We categorize a URL as fresh if the first scanned time in VirusTotal is the same as our submitted time. Almost 54% of the URLs submitted to PhishTank are categorized as fresh. For these fresh URLs, we rescan them in VirusTotal every hour and obtain a time series of scan reports. Simultaneously, we keep track of their status in PhishTank. If the URL is verified in PhishTank, we label the URL as Phishing. For malware URLs, we use URLhaus that operates as a database of malware URLs submitted by users. We collect VirusTotal scan reports for newly submitted URLhaus URLs similar to PhishTank. We observe that 60% of the URLs submitted to URLhaus are categorized as fresh.

Since PhishTank and URLhaus are scanners in VirusTotal, we exclude the responses from these scanners from the scan report to avoid biases. This dataset is used to obtain the labels for supervised and semi-supervised approaches.

Final Evaluation. We collect 3000 reports, corresponding to 3000 distinct URLs, over 7 days from the daily VirusTotal URL feed using an online stratified sampling based approach proposed in [34], [35]. The goal is to select a small set of scan reports to be labeled by human experts. These URLs (and domains) are distinct from the ones from the training

TABLE I: Datasets Utilized in Experiments

Domain	Dataset for PreTraining				Evaluation Dataset			
	#Benign Entities	#Malicious Entities	#Benign Scan Reports (Training Set)	#Malicious Scan Reports (Training Set)	#Benign Entities	#Malicious Entities	#Benign Scan Reports (Testing Set)	#Malicious Scan Reports (Testing Set)
Phishing URLs	890K	83K	1.9M	381K	2141	536	2141	536
Malicious URLs		29K		263K		239		239
Malware Files [8], [31]	7226	7197	2.8M	2.8M	236	120	18.3K	12K
IP Blacklists [19], [32]	0.97M	2.6M	13.6M	36.4M	389K	131K	8.8M	3.2M

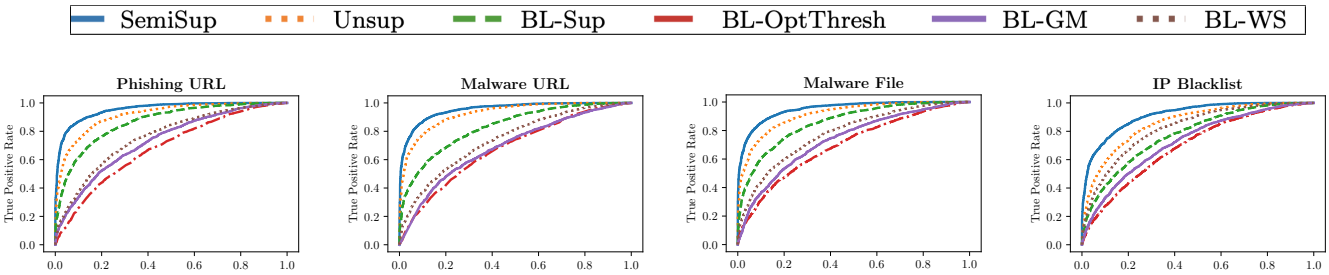


Fig. 3: Comparing the performance of SIRAJ against diverse baselines for maliciousness detection task.

data. The scan reports used for training and evaluation did not overlap temporally.

The ground truth for these scan reports is obtained through manual inspection by cyber security experts. They follow the same rubrics and look for signs of malicious URLs, including but not limited to: distribution of malware binaries, association with known indicators of compromises, domain squatting (e.g., using a popular brand name within a domain name), the presence in social engineering attacks (e.g. phishing emails), mimicking the look and feel of benign websites (e.g, login pages of banks or online stores), owner of the website, historical registration records of the website, TLS certificates associated with the website (if available), historical domain access patterns available through passive DNS services and the infrastructure in which the domain is hosted.

IP Blacklists. We use an *unlabeled* dataset collected by the authors of [19] for our experiments. Overall, they monitor 157 publicly available blacklists and cover a wide variety of attack vectors such as spam, malware, DDoS attacks, ransomware, etc. The blacklists have a variety of update frequencies ranging from 15 minutes to 7 days. We conducted our experiments over the daily snapshot of the blacklists for the years 2019-2020. Specifically, we use a form of temporal cross validation where we use six months of daily snapshots for pre-training, one month for fine-tuning and the following three months of data for evaluation. We repeat this process for every contiguous 6 month duration. For example, we first pre-train on blacklists from Jan-Jun 2019, then from Feb-Jul 2019 and so on.

We treat the output of BLAG [19] as a proxy for the ground truth BLAG achieves the state of the art results with specificity of as high as 99% and can also report malicious sources as much as 9.4 days ahead of the best blacklist. Our rationale for comparing SIRAJ against BLAG is to demonstrate that our domain agnostic pretext tasks are able to *automatically* learn the necessary knowledge and does not require expert input. In contrast, BLAG requires sophisticated aggregation and IP expansion based techniques designed by domain experts that are specific to IP domain.

B. Experimental Setup

We implement two variants of our approach. When no labeled data is available, we use the unsupervised method (UNSUP) – the clustering based approach proposed in Section VI-B. The SEMISUP variant handles the limited data scenario by building a semi-supervised model that combines both the labeled data and unlabeled data. We use a fixed labeled data size of 100 scan reports corresponding to 100 distinct entities. We implement the encoder as a three linear layer network with relu as the activation function. We use the mean squared error function for measuring reconstruction (task 1) and consistency loss (task 3). For task 2, we use the cross entropy loss function. Finally, we use the rmsprop optimizer. We use a grid search to find the optimal value of $p = 0.05$ for the hyper-parameter controlling mask corruption probability.

Baselines. We evaluate our approach against four diverse and representative baselines. Similar to SIRAJ, each of these baselines is data-driven and domain agnostic. They do not rely on domain specific features and perform the prediction based on scan reports. (i) BL-OPTTHRESH is a strengthened version of the widely used threshold-based approach that determines an entity as malicious if more than K scanners report it as malicious. Instead of assuming an ad-hoc threshold such as 1 or 3, K is chosen in an optimal manner to provide the best results for the baseline. We assume the availability of an oracle that can give us the F1-score for all possible values of K. Then, we choose K that provides the highest F1-score. (ii) BL-SUP is a supervised approach that trains a deep neural network classifier with three layers using 10% of the labeled scan reports. (iii) BL-GM is an EM-based unsupervised approach proposed in [10] for malware files. We adapt this approach for other domains by identifying the appropriate values for the latent variables of the generative

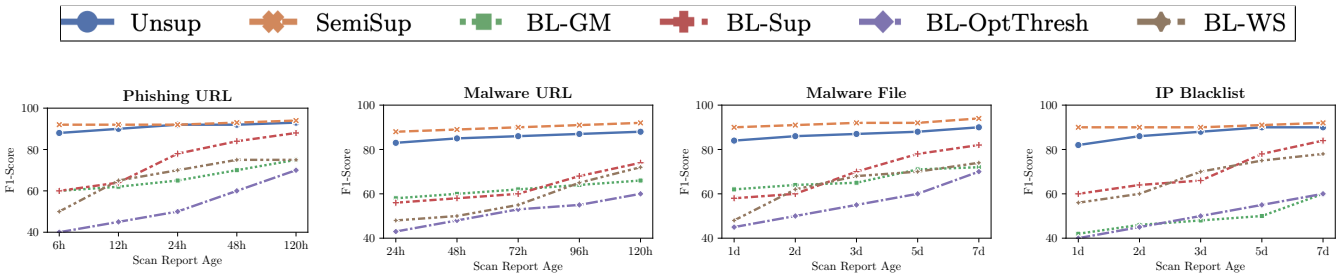


Fig. 4: Comparing the performance of our approach against diverse baselines for the early detection task.

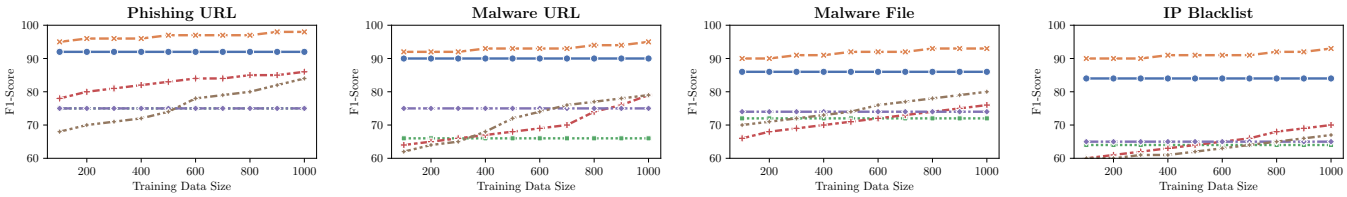


Fig. 5: Impact of Training Data Size on SIRAJ and its baselines. Legend same as that of Figure 4.

model through an exhaustive grid search that provides the best results. Finally, (iv) BL-WS is based on the paradigm of weak supervision [24]. Each scanner is modeled as a noisy classifier through a labeling function. Then, we build a generative model based on the commonalities and contradictions between the outputs of the scanners. We then train a noise-aware discriminative classifier using the same labeled data as that of BL-SUP. For additional details, please refer to [24].

Due to unbalanced nature of the datasets, we use F-score to measure the performance that balances precision and recall. It is defined as $\frac{2\times\text{precision}\times\text{recall}}{\text{precision + recall}}$.

Grouping Scan Reports. We focus on fresh files and URLs where timely detection is important. We use the term 'age' to denote the duration between the current time and when it is first seen in an aggregate service such as VirusTotal. We group the entities based on their age deciles where each decile consists of an exact 10% of the evaluation dataset. We construct an equi-depth histogram [36] by sorting the entities based on their age and the deciles correspond to the 10-th, 20-th, . . ., quantiles respectively. The early deciles corresponding to 'fresh' entities are often the most challenging.

C. Early and Accurate Malicious Entity Detection

First, we evaluate the performance of SIRAJ over two key dimensions: (a) can SIRAJ provide *accurate* predictions? and (b) can SIRAJ detect if an entity is malicious or benign *early*?

Figure 3 shows the ROC curve for SIRAJ and the baselines when evaluated on a per-entity basis. In other words, if an entity (such as an URL) has multiple temporally distinct scan reports, we only use the earliest scan report. We then use SIRAJ and the baselines to predict whether the entity is benign or malicious. This provides a realistic measure on algorithm's performance for unseen entities. Both SEMISUP and UNSUP achieve the best results outperforming a wide variety of competing baselines including BL-SUP that had as much as 10% of labeled scan reports (see Table I).

Our analysis finds that SIRAJ provides consistently high performance across the age deciles. In contrast, the other baselines perform poorly for highly fresh entities (those with early deciles), and only catch up with SIRAJ for older entities. We also find that UNSUP outperforms BL-WS which in turn outperforms BL-GM. UNSUP uses a two-step process that involves learning scanner dependencies and augmenting them with pretext tasks. In contrast, BL-WS uses a fixed generative model based on scanners' overlap and conflicts followed by a *supervised* noise-aware discriminative model using labeled data. This shows that a careful design of pretext tasks plays a major role in the outperformance of SIRAJ while also reducing the reliance on the labeled data. Finally, the superior performance of BL-WS over BL-GM shows that it is often desirable to *learn* both the structure and parameters of the generative model instead of fixing the generative model (based on domain knowledge from experts) and just learning the parameters. In fact, this design choice is what allowed SIRAJ to transfer between multiple domains seamlessly.

Comparing SIRAJ against Domain Aware Baselines. Both SIRAJ and the baselines are domain agnostic and do not make any domain specific assumptions. We also compare against a well-known domain-aware and *unsupervised* baseline. Figure 3(d) shows the performance of SIRAJ against BLAG. Recall that we used BLAG [19] as a proxy for ground truth which means that it achieves perfect performance. We can see that UNSUP and SEMISUP have a high degree of agreement with BLAG. BLAG uses sophisticated techniques such as aggregation and expansion (transforming IP addresses into IP prefixes). We find that our pretext tasks – even when not specifically designed for IP domain – are able to incorporate some of these aspects. Specifically, the generative model and pretext task 1 learn the scanner dependencies and use them to perform sophisticated aggregation. Our pretext tasks 2 and 3 produce temporally consistent embeddings that are also

IP prefix aware. We find that our encoder produces similar embeddings for scan reports of two IPs from the same block as compared to scan reports of two random IPs.

Early Detection of Malicious Entities. We next investigate the performance of the various algorithms for early detection. Given a set of entities $\mathcal{E}$ and a time duration δ, we obtain a time series of scan reports for each $e \in \mathcal{E}$ for time periods $T, T+\delta, T+2\delta, \ldots$ where T is the submitted time for entity e. For example, if δ is one day, we get daily scan reports for all entities. We then measure the F-score for the scan reports at every time snapshot. If an entity e is malicious, an algorithm should identify it as early as possible. Intuitively, this task is much more challenging as the smaller δ is, the fewer scanners would label the entity as malicious and its report also tends to change significantly in the future. In contrast, as time progresses past the label stabilization period [8] even a simple threshold-based approach could give good results.

Figure 4 shows that our proposed approaches UNSUP and SEMISUP provide consistently high performance as early as 6 hours for Phishing URLs and 24 hours for other types of entities. In contrast, the competing baselines provide poor results for the early scan reports, some of which eventually catch up when δ increases. Two key aspects of this trend are notable. The second pretext learns the temporal scanner dependencies and is responsible for the good performance as early as 6 hours for Phishing URLs. The third pretext task based on temporal consistency keeps our approach's performance consistent across time. This is achieved by ensuring that embedding of the scan reports of an entity e for two consecutive timestamps $R_e^{T_1}$ and $R_e^{T_2}$ are closer to each other.

D. Ablation Analysis

Next, we conduct a series of ablation analyses to understand the contribution of the significant components and tasks in our approach. While we present the results of SEMISUP for the limited data scenario, the results for UNSUP are similar.

Ablation of Components. SIRAJ is based on three components: generative model, pre-training using self-supervised learning, and fine-tuning. We compare three variants: SEMISUP with all the components, NOGM that removes the generative model, and NOSSL that removes all the three pretext tasks. Figure 6 shows the results of this analysis. Not surprisingly, SEMISUP provides the best results showing that one needs both generative models and pretext tasks for best performance. NOGM pays a limited penalty than NOSSL for all domains showing that self-supervised learning accounts for most of SIRAJ's performance. NOSSL achieves more inferior results for fresh entities due to the lack of pretext tasks 2 and 3 that imbues our model with temporal dynamics. The performance of both variants is comparable for older entities.

Ablation of Pretext Tasks. In our next set of experiments, we seek to understand the relative importance of the pretext tasks. Specifically, we create three variants - ONLYT1, ONLYT2 and ONLYT3. As the name suggests, these approaches use a single pretext task instead of all three. ONLYT1 seeks to learn scanner dependencies in a single-time snapshot. ONLYT2 seeks to learn the temporal dependencies between scanners and ONLYT3 seeks to ensure temporal consistency of embeddings without directly learning the dependencies. Figure 7 shows that all three pretext tasks are vital for achieving good performance and the tasks are not redundant. The best performance – especially for fresh entities – is provided by ONLYT2. This is not surprising as this variant learns the temporal dependencies that are essential for early detection. The performance of ONLYT1 and ONLYT2 eventually overlap for older entities when the temporal dependencies become less relevant. ONLYT3 performs the worst as it focuses on ensuring temporal consistency of embeddings. However, as shown in Figure 4, the steady performance of SEMISUP and UNSUP in early detection is primarily due to the third pretext task.

E. Sensitivity Analysis

Next, we investigate the robustness of SIRAJ. We report the results for SEMISUP as the results for UNSUP are similar.

Robustness to Corruption. Recall that we take the scanner report as input and use the encoder to produce the corresponding embedding. This embedding is then used for downstream tasks. The primary goal of the embedding is to incorporate the dependencies and temporal dynamics of scanners. An appealing by-product is that our encoder based approach also makes the embedding more robust to missing data and corruption. Due to limited coverage, it is not unusual that a scanner does not provide any response when queried about an entity. Even when there is a response, its marking of the entity may not be correct. All these contribute to missing data and inaccurate/corrupted scan reports. We investigate two sources of corruption – random and adversarial. In the former, a small portion of randomly chosen scanner results is converted to no response. In the latter, the scanner results are flipped in an adversarial manner by focusing on the most accurate scanners. Figure 8 shows the results for random and adversarial corruption of 5% and 10% of the results. We can see that when the corruption is random, SIRAJ's performance is not affected much. Our encoder is still able to produce appropriate embeddings that paper over this issue. However, the drop in performance is comparatively higher for adversarial corruption. This drop is especially steep for fresh entities where the responses of leading scanners are much helpful. The performance of competing baselines dropped by as much as 50% even for 5% random corruption justifying our design choice of using encoders to learn an embedding.

Robustness to Low Quality Scanners. Scanning services such as VirusTotal often have a large number of scanners with varying quality and expertise. We investigate the performance of SIRAJ when using a subset of top scanners. Specifically, we compute the relative accuracy of the scanners and identify the top-50% and top-25% scanners using a held-out auxiliary dataset that is distinct from the scan reports used for training and evaluation. Then, we re-run our experiments by subsetting the scan reports containing the reports only for these scanners. Figure 9 shows that SIRAJ performs best when the entire slate of scanners are provided, and the variants SEMISUP-50 and

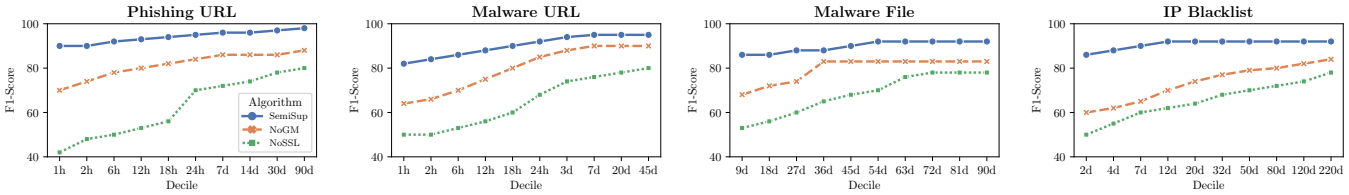


Fig. 6: Ablation analysis of the major components in our proposed approach for the detection of malicious entities task.

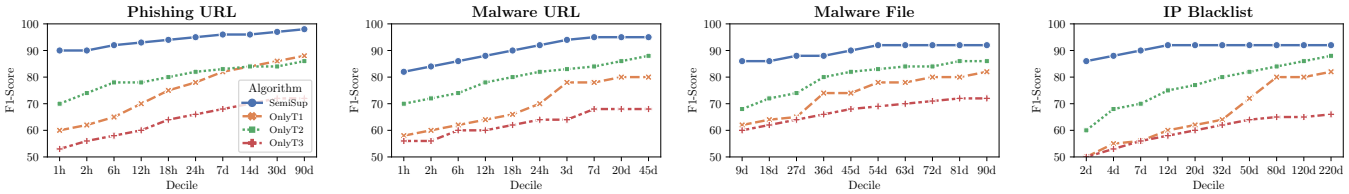


Fig. 7: Ablation analysis of the pretext tasks and on the detection of malicious entities task.

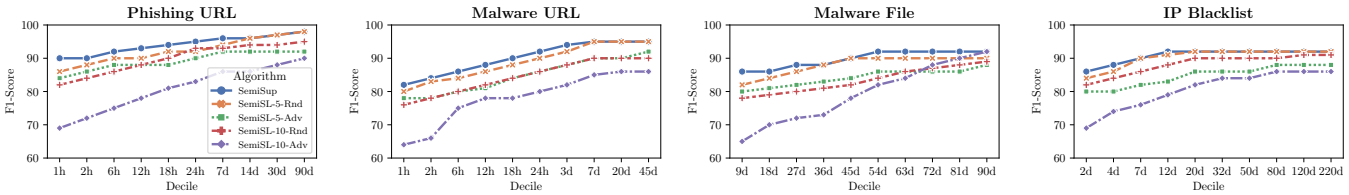


Fig. 8: Robustness of our proposed approach against random and adversarial corruption of scan reports.

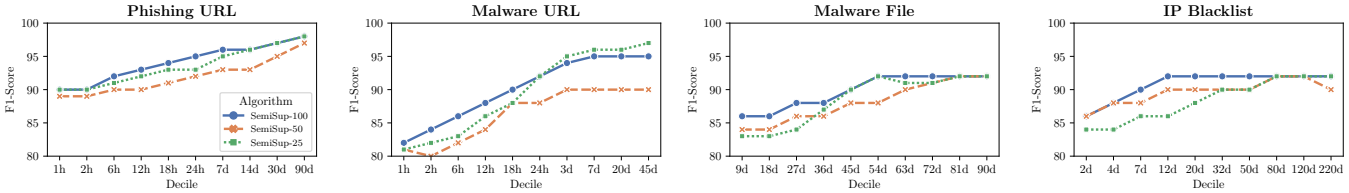


Fig. 9: Performance of our approach using a subset of scanners.

SEMISUP-25 come second depending on the domain. It shows that though low-quality scanners would inevitably introduce noise and inconsistency to scan reports, our generative model and pretext tasks could filter them out through dependency and temporal dynamic analysis and keep useful information to improve the overall performance. Due to its design, SIRAJ works well to aggregate scanners of diverse qualities.

Impact of Training Data. A key selling point of SIRAJ is its ability to achieve good performance with no (for UNSUP) and 100 (for SEMISUP) labeled scan reports. Figure 5 shows how the performance of SIRAJ and the baselines are impacted when we vary the training data from 100 to 1000 scan reports corresponding to *distinct* entities. As expected, the performance of unsupervised approaches such as UNSUP, BL-GM and BL-OPTTHRESH is unaffected. The use of embeddings allows SEMISUP to achieve good performance even with 100 scan reports that slowly improves with additional labeled data. In contrast, BL-SUP performs poorly when there is insufficient labeled data. BL-SUP trails SIRAJ even with as much as 1000 labeled scan reports. The dynamic nature of cyber security often requires periodic retraining which further exacerbates the effort needed for making BL-SUP work well. Interestingly, BL-WS outperforms BL-SUP for two domains (malware URLs and malware files) and closely trails in the other two. BL-WS uses a two-step process that learns the scanner dependencies using a generative model followed by training a noise-aware supervised model using labeled data. The outperformance shows the promise of learning dependencies using generative models (also done by SIRAJ).

F. Miscellaneous Experiments

First, we investigate the beneficial nature of our embeddings for the malware domain. While SEMISUP and UNSUP use the embeddings, the supervised variant BL-SUP uses the raw scan reports. Figure 10 shows the result of another supervised variant SUPERVISED-VEC that is trained on the embeddings instead of the raw scan reports. Since the labels for BL-SUP and SUPERVISED-VEC are identical, any performance boost is due to embeddings. SUPERVISED-VEC achieves better performance than BL-SUP especially for fresh entities as the

embeddings provide useful temporal information due to the third pretext task.

Next, we vary the underlying algorithm for semi-supervised learning. Specifically, we investigate three commonly used variants. VIME [28] is the state-of-the-art approach for self- and semi-supervised learning for the tabular domain. DeAE [37] is based on denoising autoencoders that are widely used for tabular data. Given a scan report, it learns robust features by corrupting it (similar to pretext task 1) and then forcing the autoencoder to output the denoised scan report. Intuitively, this approach allows the model to learn noise-resistant and robust features of the scan report. Finally, TabNet [38] seeks to learn salient features through sequential attention and performs unsupervised pre-training and supervised fine-tuning based on an encoder-decoder architecture. Figure 11 shows the results. SEMISUP substantially outperforms each of the other variants even though they share some design choices such as pretraining, masked learning, and so on. This shows the overall performance of SEMISUP is due to the careful construction of multiple inter-related ideas of generative models and pretext tasks. Any subset of them (as done by VIME, DeAE, and TabNet) is insufficient.

Retraining Models. Next, we investigate the impact of delay in retraining our models. Figure 12 shows that delaying the retraining has a different impact based on the domain. This is especially detrimental to the URL domain, where the shelf-life of URLs is short. The performance drops steeply after five days while the file and IP domains show a limited drop even after two weeks. Figure 13 shows the impact of the hyper-parameter δ that influences pretext tasks 2 and 3. Given a scan report R_e^t, pretext task 2 seeks to estimate $R_e^{t+\delta}$. Not surprisingly, the appropriate value depends on the domain. It is as little as 6 hours for Phishing URLs while it is as much as 168 hours for malware files. SIRAJ could be re-trained within a reasonable amount of time to cope with the dynamics of scanners. For example, SIRAJ requires 129 minutes in Tesla V100 GPU to train both encoder and classifier for 50 million scan reports for the IP blacklists.

G. Interpretability Analysis

In this section, we conduct an initial analysis to understand the source of outperformance of SIRAJ. A plausible hypothesis is that SIRAJ relies on a small subset of well-performing scanners and gives them high weights. However, this does not explain the good performance of UNSUP which does not use *any* labeled data and hence cannot automatically discern the best scanners. Furthermore, BL-SUP had access to much more labeled data than UNSUP and SEMISUP. If giving higher weights to some scanners was the secret recipe, it would have been identified by BL-SUP even if it used a simple linear classifier (instead of a powerful DL based non-linear classifier). Furthermore, modeling the scanner dependencies alone is insufficient as UNSUP (and SEMISUP) outperform BL-GM that uses a generative model to learn the dependencies. Finally, the combination of generative model and labeled data in BL-WS is also outperformed by UNSUP.

We conducted an additional investigation to understand the latent space learned by the encoder of SIRAJ. SIRAJ uses sophisticated techniques such as the generative model, multi-task learning, self- and semi-supervised learning. There has been limited prior work on interpretability analysis in each of these topics and almost none that can be used in conjunction. Hence, we consider a simplified setting by removing both the generative model and the semi-supervised learning component and focus on the pretext tasks and their impact on the encoder.

We randomly choose 5000 scan reports and compute the corresponding embeddings. Given the high dimensional nature of the embeddings, we use t-SNE [39] to reduce the dimensionality to two. Furthermore, we choose the hyperparameters of t-SNE to minimize perplexity [40]. A low-dimensional projection of the latent space can be found in Figure 14. The benign scan reports are marked in green while the malicious ones are marked in red. We observe that the encoder organizes the latent space into one or more clusters. Each cluster is relatively homogeneous consisting of either benign or malicious scan reports. This latent space also explains why SEMISUP is able to outperform BL-SUP with 100 scan reports by exploiting the cluster structure inherent in the latent space. We did not find any unifying theme explaining the clusters. The scan reports within each cluster had some common properties that did not transfer across other clusters. For example, one cluster could consist of scan reports where scanners S_i, S_j, S_k predict that the entity is malicious. Another cluster consisted of scan reports where more than l scanners changed their predictions within a time period of δ and so on. Interestingly, the cluster behavior did not persist across domains with different domains such as URLs, files and IPs having very different latent space clustering and corresponding semantics. A systematic investigation of the latent space is out-of-scope for this paper and is a promising future work.

VIII. RELATED WORK

Aggregating Security Intelligence. The most popular approach for aggregating the scan reports is the unweighted threshold strategy that marks the entity as malicious if the number of positive labels is more than a heuristically chosen threshold [6], [2], [4], [41], [42], [43]. The thresholds are arbitrarily chosen and could vary from 1 [2], [3], [4], 2 [5], [6], and 5 [7]. However, this approach is limited as it ignores different qualities of sources, including coverage and accuracy [19], [44]. There have been a few recent efforts to measure the qualities of different intelligence sources [44], [45], [29], [8], [9], [46], [11] or to smartly aggregate different sources with consideration of qualities [19], [10], [47], [48]. Ramanathan *et al.* proposed a system, BLAG, to better aggregate multiple IP blacklists for more coverage and improved accuracy by leveraging a recommender system [19]. Each of the works described focused on aggregation for a specific type of intelligence source (e.g., malware [10], [49], [47] and IP [19]). Instead, we propose a generic approach that can be applied to aggregate any type of intelligence source.

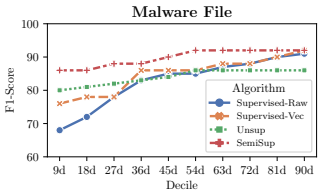


Fig. 10: Embedding Quality

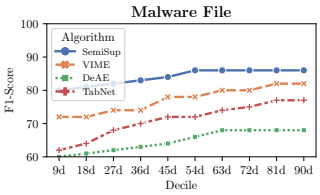


Fig. 11: Varying Semi-Supervised Learning approach

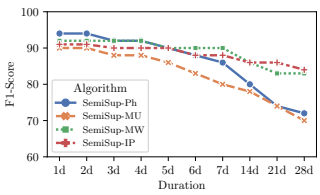


Fig. 12: Varying retraining time

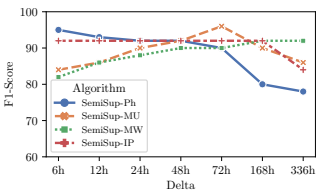
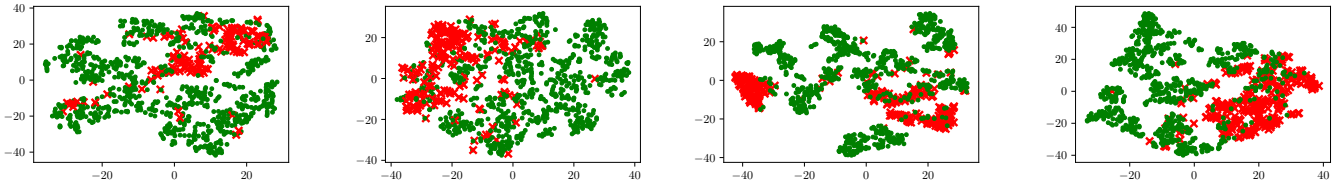


Fig. 13: Varying granularity of retraining



(a) Phishing URL (b) Malware URL (c) Malware File (d) IP Blacklist

Fig. 14: Visualization of the latent space learned by SIRAJ using the Pretext tasks.

Machine Learning based Detection of Malicious Internet Resources. Attackers increasingly utilize Internet resources for short time periods and dispose of them afterward. Hence, recent research efforts attempt to detect or predict malicious resources early. These efforts often rely on supervised machine learning algorithms with high quality ground truth on malicious URLs [12], [5], [13], [14], malicious files [15], [16], [17], [18], [50] or malicious IP addresses [19], [20] from threat intelligence reports. Other approaches rely on a threshold that results in either excessive false positives (low threshold) [18], [12] or false negatives (high threshold) [16], [50]. SIRAJ could be utilized to improve the quality of the ground truth and subsequently, the performance measurements. Recently, some efforts such as Attack2vec [51] and Log2vec [52] have used an embedding based approach for understanding the evolution of cyber attacks and detecting malicious enterprise log events recorded from attacks respectively.

Dependencies and Dynamics for Pretext Task Design. There has been extensive work on understanding the life-cycle of malicious entities and dependencies and dynamics exhibited by the scanners. These insights inspired the design of our pretext tasks. Most of the empirical analysis focuses on a specific domain such as phishing [9], [33], [53], malware [8], [54], [55], [56], and IP blacklists [57], [53], [58].

Modeling Noisy Scanners with Generative Models. Generative modeling is a principled approach for handling noisy labels used in the crowdsourcing domain, where the response of a worker for a task is considered as a noisy label [59]. Increasingly sophisticated generative models [60], [61] that consider worker and task-related parameters (such as sensitivity and specificity of the worker, task difficulty) have been proposed. Unfortunately, a direct application is not suitable for the cyber security domain as the scanners exhibit low false positive rates and high false negative rates. We are aware of just two prior works [10], [23] on generative modeling for aggregating VirusTotal scan reports. A related work Sakib *et al.* proposed a mathematical model to find the optimal combination of malware scanners for the best detection accuracy with the consideration of dependencies among scanners [49]. Our work differs in two crucial aspects. First, we do not assume that the generative model is pre-specified. Second, unlike [10], [23], we do not assume scanner outputs are conditionally independent given the maliciousness of the entity. Instead, we use a data-driven approach for learning the dependencies. Weak supervision [24] based approaches learn a specific type of generative model based on scanner overlaps and conflicts. In contrast, we learn a more expressive dependency model [26].

IX. Conclusion

We propose SIRAJ, a unified framework that can aggregate the scan reports of diverse domains such as malware files, phishing URLs, malware URLs, and malicious IPs. We adapt recent innovations in generative modeling and self-supervised learning to ensure that it can work well even when labeled data is scarce or non-existent. Another significant contribution is the design of three pretext tasks that are carefully constructed to learn scanner dependencies and their temporal dynamics. Our experimental evaluation shows that our approach works well across multiple domains and can be fine-tuned for various downstream tasks and robust to both random and adversarial corruption. SIRAJ can be used both to generate high quality ground truth for supervised approaches that detect/predict attack vectors, and to compile high-quality blacklists of attack vectors early in their life cycle to minimize the damage caused. It is our belief that our work will trigger more research in generative modeling, pretext task design, and the development of unified frameworks that can work for multiple domains instead of being siloed into individual domains.

REFERENCES

[1] VirusTotal, Subsidiary of Google. Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/. Accessed: 04-02-2021.

[2] N. Miramirkhani, T. Barron, M. Ferdman, and N. Nikiforakis, "Panning for gold.com: Understanding the dynamics of domain dropcatching," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 257–266. [Online]. Available: https://doi.org/10.1145/3178876.3186092

[3] A. Sarabi and M. Liu, "Characterizing the internet host population using deep learning: A universal and lightweight numerical embedding," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 133–146. [Online]. Available: https://doi.org/10.1145/3278532.3278545

[4] K. Tian, S. T. K. Jan, H. Hu, D. Yao, and G. Wang, "Needle in a haystack: Tracking down elite phishing domains in the wild," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 429–442. [Online]. Available: https://doi.org/10.1145/3278532.3278569

[5] M. Nabeel, I. M. Khalil, B. Guan, and T. Yu, "Following passive dns traces to detect stealthy malicious domains via graph inference," *ACM Trans. Priv. Secur.*, vol. 23, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3401897

[6] M. Sharif, J. Urakawa, N. Christin, A. Kubota, and A. Yamada, "Predicting impending exposure to malicious content from user behavior," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1487–1501.

[7] O. Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 333–343. [Online]. Available: https://doi.org/10.1145/2872427.2883056

[8] S. Zhu, J. Shi, L. Yang, B. Qin, Z. Zhang, L. Song, and G. Wang, "Measuring and modeling the label dynamics of online anti-malware engines," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[9] P. Peng, L. Yang, L. Song, and G. Wang, "Opening the blackbox of virustotal: Analyzing online phishing scan engines," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 478–485.

[10] A. Kantchelian, M. C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, A. D. Joseph, and J. D. Tygar, "Better malware ground truth: Techniques for weighting anti-virus vendor labels," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '15. New York, NY, USA: ACM, 2015, pp. 45–56. [Online]. Available: http://doi.acm.org/10.1145/2808769.2808780

[11] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, "Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 379–396.

[12] N. Miramirkhani, T. Barron, M. Ferdman, and N. Nikiforakis, "Panning for gold.com: Understanding the dynamics of domain dropcatching," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 257–266. [Online]. Available: https://doi.org/10.1145/3178876.3186092

[13] R. D. Silva, M. Nabeel, C. Elvitigala, I. Khalil, T. Yu, and C. Keppitiyagama, "Compromised or attacker-owned: A large scale classification and study of hosting domains of malicious urls," in *30th USENIX Security Symposium (USENIX Security 21)*. Vancouver, B.C.: USENIX Association, Aug. 2021.

[14] M. Sharif, J. Urakawa, N. Christin, A. Kubota, and A. Yamada, "Predicting impending exposure to malicious content from user behavior," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1487–1501. [Online]. Available: https://doi.org/10.1145/3243734.3243779

[15] B. Cheng, J. Ming, J. Fu, G. Peng, T. Chen, X. Zhang, and J.-Y. Marion, "Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 395–411. [Online]. Available: https://doi.org/10.1145/3243734.3243771

[16] D. Kim, B. J. Kwon, and T. Dumitraş, "Certified malware: Measuring breaches of trust in the windows code-signing pki," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1435–1448. [Online]. Available: https://doi.org/10.1145/3133956.3133958

[17] D. Korczynski and H. Yin, "Capturing malware propagations with code injections and code-reuse attacks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1691–1708. [Online]. Available: https://doi.org/10.1145/3133956.3134099

[18] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," in *NDSS*, 2018.

[19] S. Ramanathan, J. Mirkovic, and M. Yu, "Blag: Improving the accuracy of blacklists," in *NDSS*, 2020.

[20] D. Likhomanov and V. Poliukh, "Predicting malicious hosts by blacklisted ipv4 address density estimation," in *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2020, pp. 102–109.

[21] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[22] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[23] Y. Jiang, S. Li, and T. Li, "Em meets malicious data: A novel method for massive malware family inference," in *Proceedings of the 2020 3rd International Conference on Big Data Technologies*, 2020, pp. 74–79.

[24] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, no. 3. NIH Public Access, 2017, p. 269.

[25] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[26] S. H. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 273–282.

[27] D. Fu, M. Chen, F. Sala, S. Hooper, K. Fatahalian, and C. Ré, "Fast and three-rious: Speeding up weak supervision with triplet methods," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3280–3291.

[28] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "Vime: Extending the success of self-and semi-supervised learning to tabular domain," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[29] J. Charlton, P. Du, J.-H. Cho, and S. Xu, "Measuring relative accuracy of malware detectors in the absence of ground truth," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 450–455.

[30] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash, "Boosting self-supervised learning via knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9359–9367.

[31] "USENIX Security 2020 Artifact of Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines," https://sfzhu93.github.io/projects/vt/index.html, 2021, accessed August 2021.

[32] "NDSS 2020 Artifact of BLAG: Improving the Accuracy of Blacklists," https://steel.isi.edu/projects/BLAG/data/, 2021, accessed August 2021.

[33] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G.-J. Ahn, "Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 361–377.

[34] P. N. Bennett and V. R. Carvalho, "Online stratified sampling: evaluating classifiers at web-scale," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 1581–1584.

[35] N. Katariya, A. Iyer, and S. Sarawagi, "Active evaluation of classifiers on large datasets," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 329–338.

[36] M. Muralikrishna and D. J. DeWitt, "Equi-depth multidimensional histograms," in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988, pp. 28–36.

[37] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[38] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *arXiv preprint arXiv:1908.07442*, 2019.

[39] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[40] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-sne effectively," *Distill*, vol. 1, no. 10, p. e2, 2016.

[41] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 293–307. [Online]. Available: https://doi.org/10.1145/3278532.3278558

[42] B. Cheng, J. Ming, J. Fu, G. Peng, T. Chen, X. Zhang, and J.-Y. Marion, "Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 395–411.

[43] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1118–1129.

[44] L. Qiang, J. Zhengwei, Y. Zeming, L. Baoxu, W. Xin, and Z. Yunan, "A quality evaluation method of cyber threat intelligence in user perspective," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 269–276.

[45] P. Du, Z. Sun, H. Chen, J.-H. Cho, and S. Xu, "Statistical estimation of malware detection metrics in the absence of ground truth," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 2965–2980, 2018.

[46] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Research in Attacks, Intrusions and Defenses*, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham: Springer International Publishing, 2014, pp. 1–21.

[47] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2016, pp. 230–253.

[48] M. Hurier, G. Suarez-Tangil, S. K. Dash, T. F. Bissyandé, Y. Le Traon, J. Klein, and L. Cavallaro, "Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 425–435.

[49] M. N. Sakib, C.-T. Huang, and Y.-D. Lin, "Maximizing accuracy in multi-scanner malware detection systems," *Computer Networks*, vol. 169, p. 107027, 2020.

[50] Z. Cai and R. H. Yap, "Inferring the detection logic and evaluating the effectiveness of android anti-virus apps," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 172–182. [Online]. Available: https://doi.org/10.1145/2857705.2857719

[51] Y. Shen and G. Stringhini, "Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 905–921.

[52] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1777–1794.

[53] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Sixth Conference on Email and Anti-Spam*, 7 2009.

[54] I. Martın, J. A. Hernández, S. De Los Santos, and A. Guzmán, "Analysis and evaluation of antivirus engines in detecting android malware: A data analytics approach," in *2018 European Intelligence and Security Informatics Conference (EISIC)*, 2018, pp. 7–14.

[55] M. Vasek and T. Moore, "Empirical Analysis of Factors Affecting Malware URL Detection," in *Proceedings of the eCrime Researchers Summit*, 2013, pp. 1–9.

[56] I. Gashi, B. Sobesto, V. Stankovic, and M. Cukier, "Does malware detection improve with diverse antivirus products? an empirical study," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2013, pp. 94–105.

[57] L. Metcalf and J. M. Spring, "Blacklist ecosystem analysis: Spanning jan 2012 to jun 2014," in *Proceedings of the 2Nd ACM Workshop on Information Sharing and Collaborative Security*, ser. WISCS '15. New York, NY, USA: ACM, 2015, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/2808128.2808129

[58] A. Ramachandran, D. Dagon, , and N. Feamster, "Can DNS-Based Blacklists Keep Up with Bots?" in *Third Conference on Email and Anti-Spam*, 7 2009.

[59] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" *Proceedings of the VLDB Endowment*, vol. 10, no. 5, pp. 541–552, 2017.

[60] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy, "Supervised learning from multiple experts: whom to trust when everyone lies a bit," in *Proceedings of the 26th Annual international conference on machine learning*, 2009, pp. 889–896.

[61] P. Welinder, S. Branson, P. Perona, and S. Belongie, "The multidimensional wisdom of crowds." Neural Information Processing Systems, 2011.